DSC 102: Systems for Scalable Analytics

Programming Assignment 2: Feature Engineering & Model Selection with Apache Spark
Learning Outcomes

1. Get hands-on experience with Apache Spark
2. Get familiar with feature engineering. Learn how to transform raw data into machine-learning usable data
3. Get experience on model selection
Hardware Platform: DSMLP

1. UCSD’s Data Science & Machine Learning Platform
2. Kubernetes based
4. We have automation scripts to help you setup Spark
Hardware Platform: DSMLP

Your machine

SSH Tunnel & Port Forwarding

Login node

NFS mounted

Public dataset

Spark cluster namespace

Kubernetes

Master Running:
Jupyter Notebook, Spark driver, master, UIs

Workers Running:
Spark worker

NFS mounted to the whole namespace
AWS EMR: What is EMR

1. Amazon’s managed service for Hadoop ecosystem applications
2. Require EC2 instances to host, billed separately
1. Spark
2. Tasks
Spark: Overview

1. Spark is a general-purpose cluster computing system
2. Data processing: Spark data APIs
3. Machine learning: ML (for DataFrame) and MLlib (for RDD)
4. Graph processing: GraphX
5. Streaming data: Spark Streaming
Spark Interfaces: Overview

Languages: Scala (native), Java, **Python (PySpark)**, R, SQL

APIs: multiple choices

**RDD, DataFrame, DataSet** (only in Scala and Java), **Koalas**
Python + Spark = PySpark: Architecture

Your Python code

Spark Context
Spark APIs: Overview

1. Resilient Distributed Dataset (RDD):
   a. A distributed collection of data
   b. Oldest. The most basic abstraction of Spark
   c. ML support: MLlib. Maintenance only, i.e., no new features

2. DataFrame
   a. Named columns, like a table
   b. New. Recommended API
   c. ML support: ML. Active being developed

3. Koalas
   a. Pandas API on Spark. DataFrame under the hood
   b. Even newer. Not part of Spark but also from DataBricks
   c. ML support: MLFlow, a product from DataBricks
Spark APIs: Things in Common

1. Immutable. No in-place operations. Koalas mimics mutability but is still immutable in its core.
2. Distributed
3. Lazy. No computation happens until an action is called.
   Examples:
   a. collect()
   b. count()
   c. take(), show(), head(), first()
### Spark APIs: Differences

<table>
<thead>
<tr>
<th></th>
<th>RDD</th>
<th>DataFrame</th>
<th>Koalas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction level</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Named columns</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Query optimizations</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Programming style</td>
<td>map-reduce</td>
<td>SQL</td>
<td>SQL, Pandas</td>
</tr>
</tbody>
</table>
| Best suited for        | 1. Unstructured data  
2. Low-level ops  
3. Func. Programming / Map-reduce lovers  
   Programming / Map-reduce lovers |
|                        | 1. Structured data  
2. High-level ops with optimizations  
3. Python, R, SQL users               |
|                        | 1. Pandas/Dask users who just switched to Spark |

No need to stick to one specific API. Spark offers methods to convert from one to another.
Spark APIs: Example

Two-column table: (key, value)
Task: group by key and calculate mean values.
value contains NULLs.
Ignore the NULLs.
Spark APIs: Example - Koalas and DataFrame

# Koalas
kdf = kdf.groupby('key').agg({'value': 'mean'})

# Spark DataFrame
sdf = sdf.groupBy('key').agg(F.avg('value'))
Spark APIs: Example - RDD

```python
# RDD
agg = rdd.
    filter(lambda x: x[1] is not None).
aggregateByKey((0, 0),
    lambda acc, x: (acc[0] + x, acc[1] + 1),
    lambda acc1, acc2: (
        acc1[0] + acc2[0],
    ))
)
mean = agg.mapValues(lambda x: x[0] / x[1])
rdd = rdd.map(lambda x: x[0]).
    distinct().leftOuterJoin(mean)
```
1. Spark

2. Tasks
Tasks: overview

You will be given 8 tasks.

1. Part 1 (feature engineering): 1-6
2. Part 2 (model selection): 7-8
3. For each task you need to implement a function named task_i(), the signature and return values of which is fixed. Do not modify the input/output blocks
4. For each task you will need to populate a python dict inside the function body. Use only native python data types and structures for this dict
Tasks: datasets

You will be given 5 tables from the Amazon review dataset.

1. review. The review scores for products
2. product. The metadata about each product
3. product_processed. Same as above, but with some preprocessing
4. ml_features_train. Table with all features after PA1 and PA2 part 1
5. ml_features_test. Same as above, test split
Tasks: Example

1. Description: calculate the mean of a column x
2. Output schema:
   - mean_x: float — mean of x

3. Code example:
   ```python
def task_0(df):
    res = {'mean_x': None}
    # Your implementation:------
    mean_x = df['x'].mean()
    # Put result:------
    res['mean_x'] = mean_x
    # --------------------------
    return res

You can only code in this section!
Grading

1. 100 for correctness, however, we have a timeout
2. 10 extra for runtime
Grading - Correctness

1. Several sub-tests for each task with hidden datasets
2. Your code must pass all sub-tests to be counted as passed for that task
Grading - Timing

1. Hard timeout: 2hr. Will kill after it
2. You get points for each task you passed in 2hr
3. We will deduct points if runs for more than 1hr
4. Extra points if your code is correct and runs fast
5. Details in the PDF document provided to you
Live demo time!
Million Dollar Question:
To cache or not to cache
Million Dollar Question:
Number of partitions